Apppendix
SN 08/738,659

"TCP/IP Illustrated," Vol. 1 W. Richard Stevens 1994, Addison-Wesley, pp. 441-452

28

SMTP: Simple Mail Transfer Protocol

Introduction

Electronic mail (e-mail) is undoubtedly one of the most popular applications. [Caceres et al. 1991] shows that about one-half of all TCP connections are for the *Simple Mail Transfer Protocol*, SMTP. (On a byte count basis, FTP connections carry more data.) [Paxson 1993] found that the average mail message contains around 1500 bytes of data, but some messages contain megabytes of data, because electronic mail is sometimes used to send files.

Figure 28.1 shows an outline of e-mail exchange using TCP/IP.

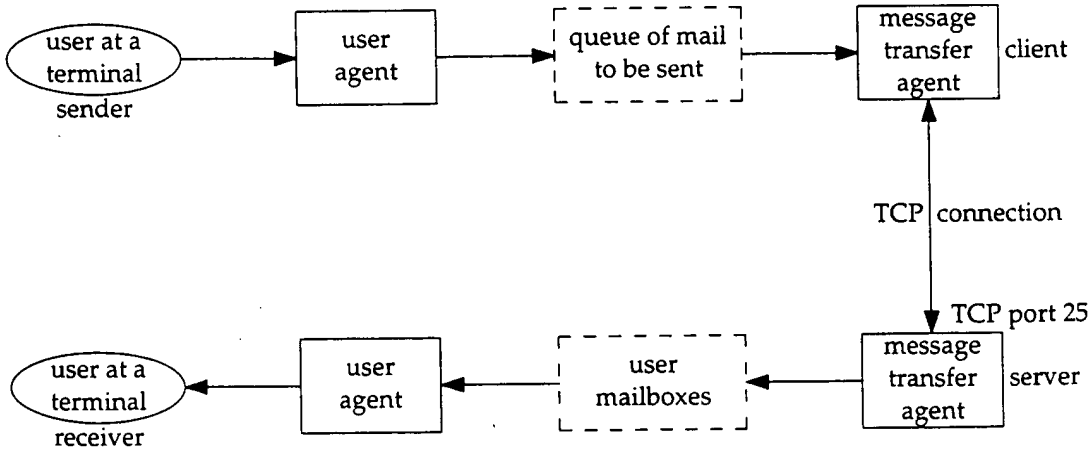

Figure 28.1 Outline of Internet electronic mail.

Users deal with a *user agent*, of which there are a multitude to choose from. Popular user agents for Unix include MH, Berkeley Mail, Elm, and Mush.

The exchange of mail using TCP is performed by a *message transfer agent* (MTA). The most common MTA for Unix systems is Sendmail. Users normally don't deal with the MTA. It is the responsibility of the system administrator to set up the local MTA. Users often have a choice, however, for their user agent.

This chapter examines the exchange of electronic mail between the two MTAs using TCP. We do not look at the operation or design of user agents.

RFC 821 [Postel 1982] specifies the SMTP protocol. This is how two MTAs communicate with each other across a single TCP connection. RFC 822 [Crocker 1982] specifies the format of the electronic mail message that is transmitted using RFC 821 between the two MTAs.

28.2 SMTP Protocol

The communication between the two MTAs uses NVT ASCII. Commands are sent by the client to the server, and the server responds with numeric reply codes and optional human-readable strings. This is similar to what we saw with FTP in the previous chapter.

There are a small number of commands that the client can send to the server: less than a dozen. (By comparison, FTP has more than 40 commands.) Rather than describing each one, we'll start with a simple example to show what happens when we send mail.

Simple Example

We'll send a simple one-line message and watch the SMTP connection. We invoke our user agent with the -v flag, which is passed to the mail transport agent (Sendmail in this case). This MTA displays what is sent and received across the SMTP connection when this flag is specified. Lines beginning with >>> are commands sent by the SMTP client, and lines beginning with a 3-digit reply code are from the SMTP server. Here is the interactive session:

```
sun % mail -v rstevens@noao.edu          invoke our user agent
To: rstevens@noao.edu                    this is output by user agent
Subject: testing                         we're then prompted for a subject
                                         user agent adds one blank line between headers and body
1, 2, 3.                                 this is what we type as the body of the message
.                                        we type a period on a line by itself to say we're done

Sending letter ... rstevens@noao.edu...  verbose output from user agent
                                         following is output by MTA (Sendmail)
Connecting to mailhost via ether...
Trying 140.252.1.54...  connected.
220 noao.edu Sendmail 4.1/SAG-Noao.G89 ready at Mon, 19 Jul 93 12:47:34 MST

>>> HELO sun.tuc.noao.edu.
250 noao.edu Hello sun.tuc.noao.edu., pleased to meet you

>>> MAIL From:<rstevens@sun.tuc.noao.edu>
250 <rstevens@sun.tuc.noao.edu>... Sender ok
```

```
>>> RCPT To:<rstevens@noao.edu>
250 <rstevens@noao.edu>... Recipient ok

>>> DATA
354 Enter mail, end with "." on a line by itself

>>> .
250 Mail accepted

>>> QUIT
221 noao.edu delivering mail

rstevens@noao.edu... Sent
sent.
```

this is output by user agent

Only five SMTP commands are used to send the mail: HELO, MAIL, RCPT, DATA, and QUIT.

We type `mail` to invoke our user agent. We're then prompted for a subject, and after typing that, we type the body of the message. Typing a period on a line by itself completes the message and the user agent passes the mail to the MTA for delivery.

The client does the active open to TCP port 25. When this returns, the client waits for a greeting message (reply code 220) from the server. This server's response must start with the fully qualified domain name of the server's host: `noao.edu` in this example. (Normally the text that follows the numeric reply code is optional. Here the domain name is required. The text beginning with `Sendmail` is optional.)

Next the client identifies itself with the HELO command. The argument must be the fully qualified domain name of the client host: `sun.tuc.noao.edu`.

The MAIL command identifies the originator of the message. The next command, RCPT, identifies the recipient. More than one RCPT command can be issued if there are multiple recipients.

The contents of the mail message are sent by the client using the DATA command. The end of the message is specified by the client sending a line containing just a period. The final command, QUIT, terminates the mail exchange.

Figure 28.2 is a time line of the SMTP connection between the sender SMTP (the client) and the receiver SMTP (the server). We have removed the connection establishment and termination, and the window size advertisements.

The amount of data we typed to our user agent was a one-line message ("1, 2, 3."), yet 393 bytes of data are sent in segment 12. The following 12 lines comprise the 393 bytes that are sent by the client:

```
Received: by sun.tuc.noao.edu. (4.1/SMI-4.1)
        id AA00502; Mon, 19 Jul 93 12:47:32 MST
Message-Id: <9307191947.AA00502@sun.tuc.noao.edu.>
From: rstevens@sun.tuc.noao.edu (Richard Stevens)
Date: Mon, 19 Jul 1993 12:47:31 -0700
Reply-To: rstevens@noao.edu
X-Phone: +1 602 676 1676
X-Mailer: Mail User's Shell (7.2.5 10/14/92)
To: rstevens@noao.edu
Subject: testing

1, 2, 3.
```

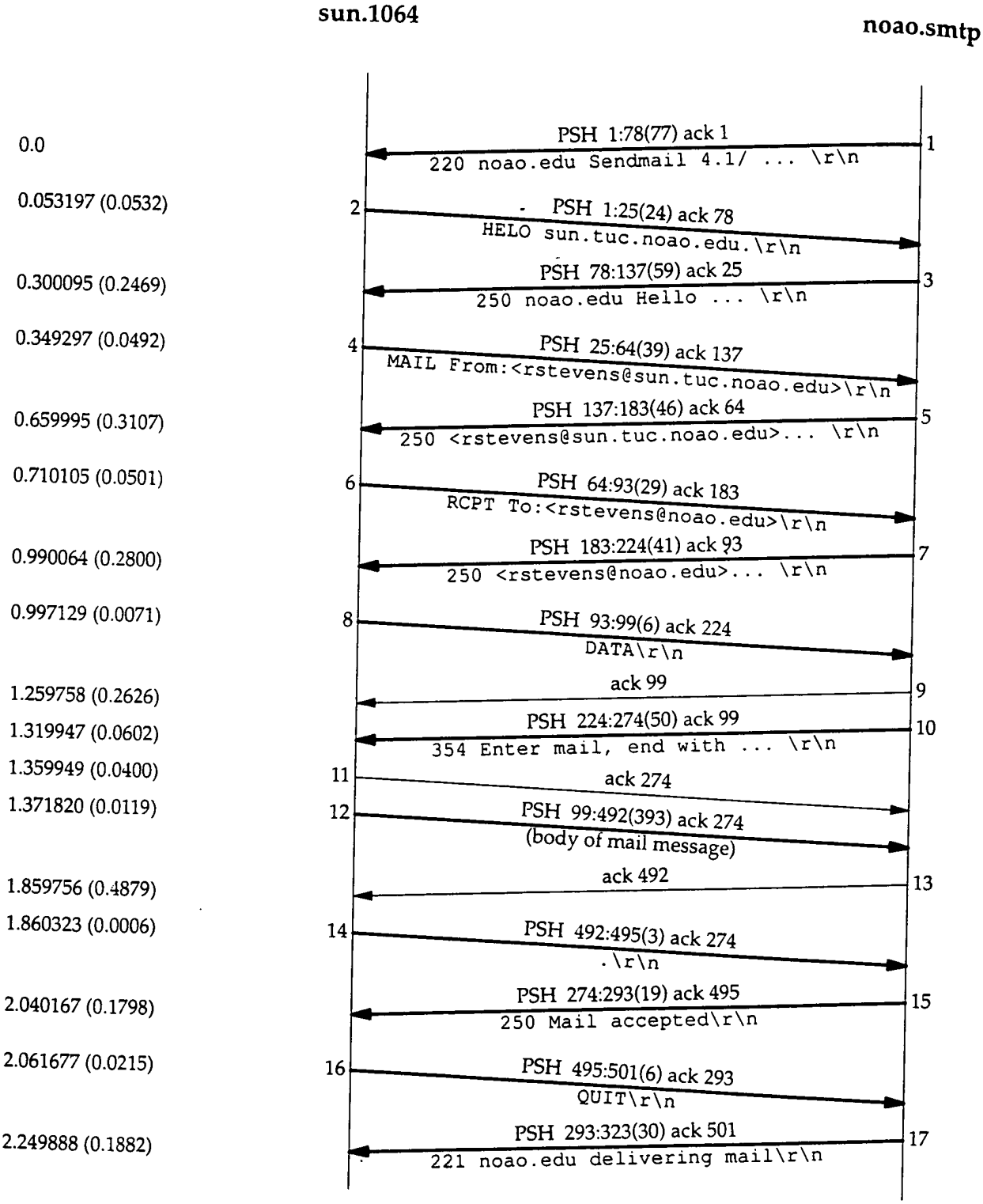

Figure 28.2 Basic SMTP mail delivery.

The first three lines, `Received:` and `Message-Id:`, are added by the MTA, and the next nine are generated by the user agent.

SMTP Commands

The minimal SMTP implementation supports eight commands. We saw five of them in the previous example: HELO, MAIL, RCPT, DATA, and QUIT.

The RSET command aborts the current mail transaction and causes both ends to reset. Any stored information about sender, recipients, or mail data is discarded.

The VRFY command lets the client ask the sender to verify a recipient address, without sending mail to the recipient. It's often used by a system administrator, by hand, for debugging mail delivery problems. We'll show an example of this in the next section.

The NOOP command does nothing besides force the server to respond with an OK reply code (200).

There are additional, optional commands. EXPN expands a mailing list, and is often used by the system administrator, similar to VRFY. Indeed, most versions of Sendmail handle the two identically.

> Version 8 of Sendmail in 4.4BSD no longer handles the two identically. VRFY does not expand aliases and doesn't follow `.forward` files.

The TURN command lets the client and server switch roles, to send mail in the reverse direction, without having to take down the TCP connection and create a new one. (Sendmail does not support this command.) There are three other commands (SEND, SOML, and SAML), which are rarely implemented, that replace the MAIL command. These three allow combinations of the mail being delivered directly to the user's terminal (if logged in), or sent to the recipient's mailbox.

Envelopes, Headers, and Body

Electronic mail is composed of three pieces.

1. The *envelope* is used by the MTAs for delivery. In our example the envelope was specified by the two SMTP commands:

   ```
   MAIL From:<rstevens@sun.tuc.noao.edu>
   RCPT To:<rstevens@noao.edu>
   ```

   RFC 821 specifies the contents and interpretation of the envelope, and the protocol used to exchange mail across a TCP connection.

2. *Headers* are used by the user agents. We saw nine header fields in our example: `Received`, `Message-Id`, `From`, `Date`, `Reply-To`, `X-Phone`, `X-Mailer`, `To`, and `Subject`. Each header field contains a name, followed by a colon, followed by the field value. RFC 822 specifies the format and interpretation of the header fields. (Headers beginning with an `X-` are user-defined fields. The others are defined by RFC 822.) Long header fields, such as `Received` in the example, are folded onto multiple lines, with the additional lines starting with white space.

3. The *body* is the content of the message from the sending user to the receiving user. RFC 822 specifies the body as lines of NVT ASCII text. When transferred

using the DATA command, the headers are sent first, followed by a blank line, followed by the body. Each line transferred using the DATA command must be less than 1000 bytes.

The user agent takes what we specify as the body, adds some headers, and passes the result to the MTA. The MTA adds a few headers, adds the envelope, and sends the result to another MTA.

The term *content* is often used to describe the combination of headers and the body. The content is sent by the client with the DATA command.

Relay Agents

The first line of informational output by our local MTA in our example is "Connecting to mailhost via ether." This is because the author's system has been configured to send all nonlocal outgoing mail to a relay machine for delivery.

This is done for two reasons. First, it simplifies the configuration of all MTAs other than the relay system's MTA. (Configuring an MTA is not simple, as anyone who has ever worked with Sendmail can attest to.) Second, it allows one system at an organization to act as the mail hub, possibly hiding all the individual systems.

In this example the relay system has a hostname of `mailhost` in the local domain (`.tuc.noao.edu`) and all the individual systems are configured to send their mail to this host. We can execute the `host` command to see how this name is defined to the DNS:

```
sun % host mailhost
mailhost.tuc.noao.edu    CNAME    noao.edu          canonical name
noao.edu                 A        140.252.1.54      its real IP address
```

If the host used as the relay changes in the future, only its DNS name need change—the mail configuration of all the individual systems does not change.

Most organizations are using relay systems today. Figure 28.3 is a revised picture of Internet mail (Figure 28.2), taking into account that both the sending host and the final receiving host probably use a relay host.

In this scenario there are four MTAs between the sender and receiver. The local MTA on the sender's host just delivers the mail to its relay MTA. (This relay MTA could have a hostname of `mailhost` in the organization's domain.) This communication uses SMTP across the organization's local internet. The relay MTA in the sender's organization then sends the mail to the receiving organization's relay MTA across the Internet. This other relay MTA then delivers the mail to the receiver's host, by communication with the local MTA on the receiver's host. All the MTAs in this example use SMTP, although the possibility exists for other protocols to be used.

NVT ASCII

One feature of SMTP is that it uses NVT ASCII for everything: the envelope, the headers, and the body. As we said in Section 26.4, this is a 7-bit character code, transmitted as 8-bit bytes, with the high-order bit set to 0.

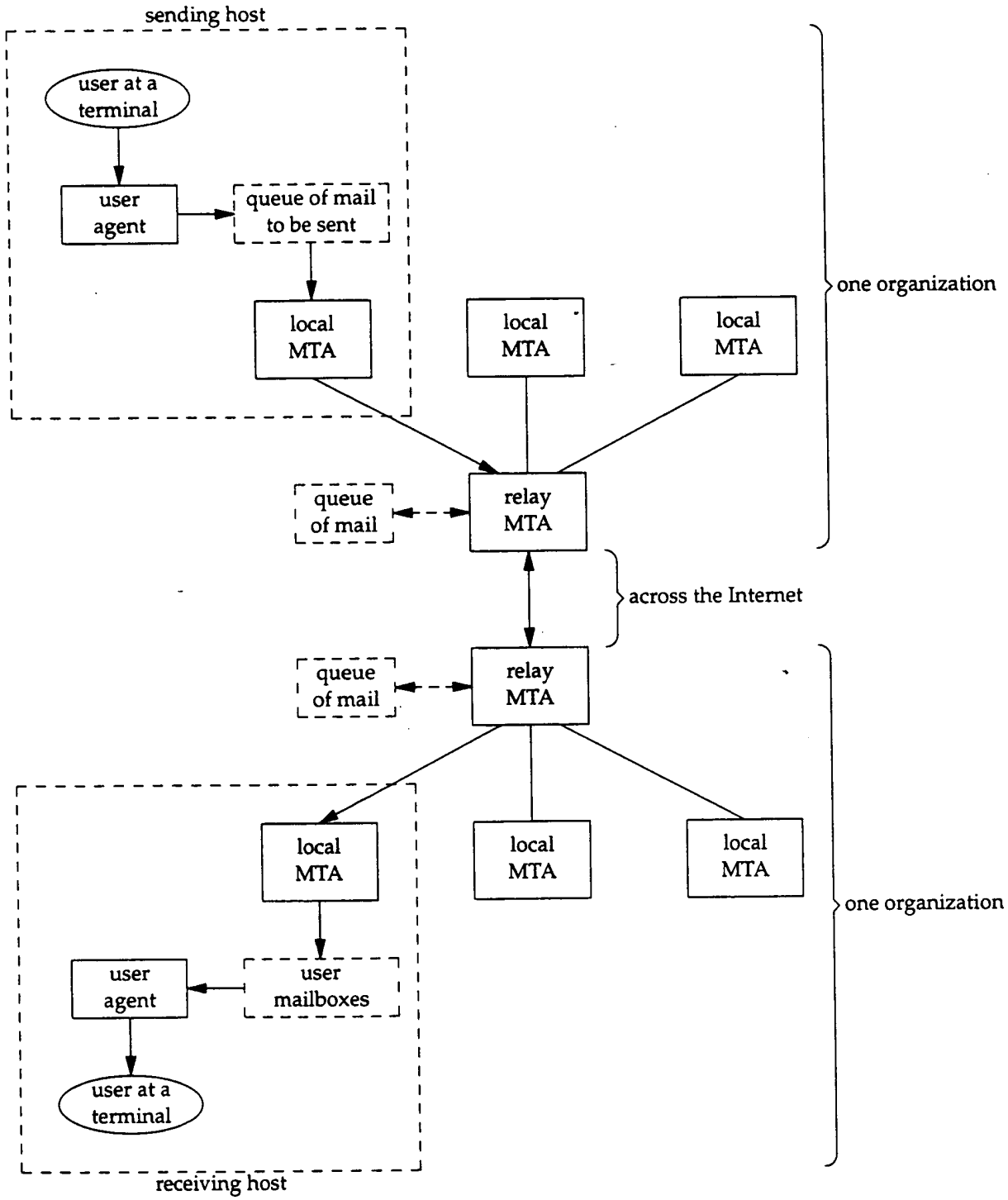

Figure 28.3 Internet electronic mail, with a relay system at both ends.

In Section 28.4 we discuss some newer features of Internet mail, extended SMTP and multimedia mail (MIME), that allow the sending and receiving of data such as audio and video. We'll see that MIME works with NVT ASCII for the envelope, headers, and body, with changes required only in the user agents.

Retry Intervals

When a user agent passes a new mail message to its MTA, delivery is normally attempted immediately. If the delivery fails, the MTA must queue the message and try again later.

The Host Requirements RFC recommends an initial timeout of at least 30 minutes. The sender should not give up for at least 4–5 days. Furthermore, since delivery failures are often transient (the recipient has crashed or there is a temporary loss of network connectivity), it makes sense to try two connection attempts during the first hour that the message is in the queue.

28.3 SMTP Examples

We showed normal mail delivery in the previous section, so here we'll show how MX records are used for mail delivery, and illustrate the VRFY and EXPN commands.

MX Records: Hosts Not Directly Connected to the Internet

In Section 14.6 we mentioned that one type of resource record in the DNS is the mail exchange record, called MX records. In the following example we'll show how MX records are used to send mail to hosts that are not directly connected to the Internet. RFC 974 [Partridge 1986] describes the handling of MX records by MTAs.

The host `mlfarm.com` is not directly connected to the Internet, but has an MX record that points to a mail forwarder that is on the Internet:

```
sun % host -a -v -t mx mlfarm.com
The following answer is not authoritative:
mlfarm.com              86388   IN   MX   10 mercury.hsi.com
mlfarm.com              86388   IN   MX   15 hsi86.hsi.com
Additional information:
mercury.hsi.com         86388   IN    A   143.122.1.91
hsi86.hsi.com          172762   IN    A   143.122.1.6
```

There are two MX records, each with a different preference. We expect the MTA to start with the lower of the two preference values.

The following script shows mail being sent to this host:

```
sun % mail -v ron@mlfarm.com          -v flag to see what the MTA does
To: ron@mlfarm.com
Subject: MX test message

                                      the body of the message is typed here (not shown)
.                                     period on a line by itself to terminate message
Sending letter ... ron@mlfarm.com...
Connecting to mlfarm.com via tcp...
mail exchanger is mercury.hsi.com     the MX records are found
Trying 143.122.1.91...  connected.   first tries the one with lower preference
220 mercury.hsi.com ...
                                      remainder is normal SMTP mail transfer
```

We can see in this output that the MTA discovered that the destination host had an MX record and used the MX record with the lowest preference value.

Before running this example from the host sun, it was configured not to use its normal relay host, so we could see the mail exchange with the destination host. It was also configured to use the name server on the host noao.edu (which is across its dialup SLIP link), so we could capture both the mail transfer and the DNS traffic using tcpdump on the SLIP link. Figure 28.4 shows the starting portion of the tcpdump output.

```
1  0.0                 sun.1624 > noao.edu.53: 2+ MX? mlfarm.com. (28)
2  0.445572 (0.4456)   noao.edu.53 > sun.1624: 2* 2/0/2 MX
                                                mercury.hsi.com. 10 (113)

3  0.505739 (0.0602)   sun.1143 > mercury.hsi.com.25: S 1617536000:1617536000(0)
                                                       win 4096
4  0.985428 (0.4797)   mercury.hsi.com.25 > sun.1143: S 1832064000:1832064000(0)
                                                       ack 1617536001 win 16384
5  0.986003 (0.0006)   sun.1143 > mercury.hsi.com.25: . ack 1 win 4096
6  1.735360 (0.7494)   mercury.hsi.com.25 > sun.1143: P 1:90(89) ack 1 win 16384
```

Figure 28.4 Sending mail to a host that uses MX records.

In line 1 the MTA queries its name server for an MX record for mlfarm.com. The plus sign following the 2 means the recursion-desired flag is set. The response in line 2 has the authoritative bit set (the asterisk following the 2) and contains 2 answer RRs (the two MX host names), 0 authority RRs, and 2 additional RRs (the IP addresses of the two hosts).

In lines 3–5 a TCP connection is established with the SMTP server on the host mercury.hsi.com. The server's initial 220 response is shown in line 6.

Somehow the host mercury.hsi.com must deliver this mail message to the destination, mlfarm.com. The UUCP protocols are a popular way for a system not connected to the Internet to exchange mail with its MX site.

In this example the MTA asks for an MX record, gets a positive result, and sends the mail. Unfortunately the interaction between an MTA and the DNS can differ between implementations. RFC 974 specifies that an MTA should ask for MX records first, and if none are found, attempt delivery to the destination host (i.e., ask the DNS for an A record for the host, for its IP address). MTAs must also deal with CNAME records in the DNS (canonical names).

As an example, if we send mail to rstevens@mailhost.tuc.noao.edu from a BSD/386 host, the following steps are executed by the MTA (Sendmail).

1. Sendmail asks the DNS for CNAME records for mailhost.tuc.noao.edu. We see that a CNAME record exists:

```
sun % host -t cname mailhost.tuc.noao.edu
mailhost.tuc.noao.edu    CNAME    noao.edu
```

2. A DNS query is issued for CNAME records for noao.edu and the response says none exist.

3. Sendmail then asks the DNS for MX records for noao.edu and gets one MX record:

```
sun % host -t mx noao.edu
noao.edu                MX      noao.edu
```

4. Sendmail queries the DNS for an A record (IP address) for noao.edu and gets back the value 140.252.1.54. (This A record was probably returned by the name server for noao.edu as an additional RR with the MX reply in step 3.)
5. An SMTP connection is initiated to 140.252.1.54 and the mail is sent.

A CNAME query is not tried for the data returned in the MX record (noao.edu). The data in the MX record cannot be an alias—it must be the name of a host that has an A record.

> The version of Sendmail distributed with SunOS 4.1.3 that uses the DNS only queries for MX records, and gives up if an MX record isn't found.

MX Records: Hosts That Are Down

Another use of MX records is to provide an alternative mail receiver when the destination host is down. If we look at the DNS entry for our host sun we see that it has two MX records:

```
sun % host -a -v -t mx sun.tuc.noao.edu
sun.tuc.noao.edu        86400   IN      MX       0 sun.tuc.noao.edu
sun.tuc.noao.edu        86400   IN      MX      10 noao.edu
Additional information:
sun.tuc.noao.edu        86400   IN       A      140.252.1.29
sun.tuc.noao.edu        86400   IN       A      140.252.13.33
noao.edu                86400   IN       A      140.252.1.54
```

The MX record with the lowest preference indicates that direct delivery to the host itself should be tried first, and the next preference is to deliver the mail to the host noao.edu.

In the following script we send mail to ourself at the host sun.tuc.noao.edu, from the host vangogh.cs.berkeley.edu, after turning off the destination's SMTP server. When a connection request arrives for port 25, TCP should respond with an RST, since no process has a passive open pending for that port.

```
vangogh % mail -v rstevens@sun.tuc.noao.edu
A test to a host that's down.
.
EOT
rstevens@sun.tuc.noao.edu... Connecting to sun.tuc.noao.edu. (smtp)...
rstevens@sun.tuc.noao.edu... Connecting to noao.edu. (smtp)...
220 noao.edu ...
                                        remainder is normal SMTP mail transfer
```

We see that the MTA tries to contact sun.tuc.noao.edu and then gives up and contacts noao.edu instead.

Figure 28.5 is the tcpdump output that shows that TCP responds to the incoming SYNs with an RST.

```
1 0.0                  vangogh.3873 > 140.252.1.29.25: S 2358303745:2358303745(0) ...
2 0.000621 (0.0006) 140.252.1.29.25 > vangogh.3873: R 0:0(0) ack 2358303746 win 0

3 0.300203 (0.2996) vangogh.3874 > 140.252.13.33.25: S 2358367745:2358367745(0) ...
4 0.300620 (0.0004) 140.252.13.33.25 > vangogh.3874: R 0:0(0) ack 2358367746 win 0
```

Figure 28.5 Attempt to connect to an SMTP server that is not running.

In line 1 vangogh sends a SYN to port 25 at the primary IP address for sun: 140.252.1.29. This is rejected in line 2. The SMTP client on vangogh then tries the next IP address for sun: 140.252.13.33 (line 3), and it also causes an RST to be returned (line 4).

The SMTP client doesn't try to differentiate between the different error returns from its active open on line 1, which is why it tries the other IP address on line 2. If the error had been something like "host unreachable" for the first attempt, it's possible that the second attempt could work.

If the reason the SMTP client's active open fails is because the server host is down, we would see the client retransmit the SYN to IP address 140.252.1.29 for a total of 75 seconds (similar to Figure 18.6), followed by the client sending another three SYNs to IP address 140.252.13.33 for another 75 seconds. After 150 seconds the client would move on to the next MX record with the higher preference.

VRFY and EXPN Commands

The VRFY command verifies that a recipient address is OK, without actually sending mail. EXPN is intended to expand a mailing list, without sending mail to the list. Many SMTP implementations (such as Sendmail) consider the two the same, but we mentioned that newer versions of Sendmail do differentiate between the two.

As a simple test we can connect to a newer version of Sendmail and see the difference. (We have removed the extraneous Telnet client output.)

```
sun % telnet vangogh.cs.berkeley.edu 25
220-vangogh.CS.Berkeley.EDU Sendmail 8.1C/6.32 ready at Tue, 3 Aug 1993 14:
59:12 -0700
220 ESMTP spoken here

helo bsdi.tuc.noao.edu
250 vangogh.CS.Berkeley.EDU Hello sun.tuc.noao.edu [140.252.1.29], pleased
to meet you

vrfy nosuchname
550 nosuchname... User unknown

vrfy rstevens
250 Richard Stevens <rstevens@vangogh.CS.Berkeley.EDU>

expn rstevens
250 Richard Stevens <rstevens@noao.edu>
```

First notice that we purposely typed the wrong hostname on the HELO command: bsdi instead of sun. Most SMTP servers take the IP address of the client and perform

a DNS pointer query (Section 14.5) and compare the hostnames. This allows the server to log the client connection based on the IP address, not the name that a user might have mistyped. Some servers respond with humorous messages, such as "You are a charlatan," or "why do you call yourself ...". We see in this example that this server just prints our real domain name from the pointer query along with our IP address.

We then type a VRFY command for an invalid name, and the server responds with a 550 error. Next we type a valid name, and the server responds with the username on the local host. Next we try the EXPN command and get a different response. The EXPN command determines that the mail for this user is being forwarded, and prints the forwarding address.

Many sites disable the VRFY and EXPN commands, sometimes for privacy, and sometimes in the belief that it's a security hole. For example, we can try these commands with the SMTP server at the White House:

```
sun % telnet whitehouse.gov 25
220 whitehouse.gov SMTP/smap Ready.

helo sun.tuc.noao.edu
250 (sun.tuc.noao.edu) pleased to meet you.

vrfy clinton
500 Command unrecognized

expn clinton
500 Command unrecognized
```

28.4 SMTP Futures

Changes are taking place with Internet mail. Recall the three pieces that comprise Internet mail: the envelope, headers, and body. New SMTP commands are being added that affect the envelope, non-ASCII characters can be used in the headers, and structure is being added to the body (MIME). In this section we consider the extensions to each of these three pieces in order.

Envelope Changes: Extended SMTP

RFC 1425 [Klensin et al. 1993a] defines the framework for adding extensions to SMTP. The result is called *extended SMTP* (ESMTP). As with other new features that we've described in the text, these changes are being added in a backward compatible manner, so that existing implementations aren't affected.

A client that wishes to use the new features initiates the session with the server by issuing a EHLO command, instead of HELO. A compatible server responds with a 250 reply code. This reply is normally multiline, with each line containing a keyword and an optional argument. These keywords specify the SMTP extensions supported by the server. New extensions will be described in an RFC and will be registered with the IANA. (In a multiline reply all lines except the last have a hyphen after the numeric reply code. The last line has a space after the numeric reply code.)